1 Introduction

In the capstone project, we performed EDA (Exploratory Data Analysis) on the data by using some python libraries (Pandas, numpy, matplotlib and seaborn). This Yellow Taxi trip data taken from New York city. There are many field in the data.

1.1 Import Libraries

Here, we use some python libraries for manipulation of data. These libraries make easy for manipulation and analysis of the data which are as given below-

- **Pandas -** Pandas library is used for manipulation of data.
- **Numpy -** Numpy library is used to perform on matrix or array and fast execution code
- **Matplotlib -** This library is used for data visualization.
- **Seaborn -** Seaborn is also used for data visualization with more attractive and effecftive way

In [ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecat
ed. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

1.2 Load data

I have **Yello Taxi Trip Records Data** in csv file (comma seprated file or flat file). This file is look like excel sheet but there are only **one sheet** in csv file.

Here, we are load the data using pandas method pd.read_csv() which is used to extract the knowledge or useful information or EDA.

```
In [ ]: data = pd.read_csv('/content/drive/My Drive/Capstone Project/yellow_tripdata_2019-12.csv')
```

```
/usr/local/lib/python3.6/dist-packages/IPython/core/interactiveshell.py:2718: DtypeWarning: Columns (6) have mixed type
s.Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

There are many attributes or fields of Yello Taxi Trip Record data as follow

- **VendorID -** The venfor ID
- **tpep_pickup_datetime -** Pickup date and time
- **tpep_dropoff_datetime -** Drop off date and time
- **passenger_count -** Number of passenger at a time
- **trip_distance -** Distance between pickup and dropoff location
- **RatecodeID -** The rate code ID
- **store_and_fwd_flag -** Store and forword flag
- **PULocationID -** Pickup location ID
- **DOLocationID -** Dropoff location ID
- **payment_type -** Payment method
- **fare_amount -** Fare amount of the trip
- **extra -** Extra charges
- **mta_tax -** MTA Tax
- **tip_amount -** Tip amount
- **tolls_amount -** Toll amount
- **improvement_surcharge -** Improvement surcharge
- **total_amount -** Total amount of the trip
- **congestion_surcharge -** Congestion surcharge

1.3 Basic information about the data

Here, I extract the basic information about the data.

- Number of rows
- Number of columns
- Dimension of the data
- Size of the data

- Shape of data
- Columns name
- Index range
- Data type of columns

```
In [ ]: print('Number of rows- ', data.shape[0])
        print('Number of columns- ',data.shape[1])
        print('size of data- ',data.size)
        print('Dimension of data- ',data.ndim)
```

```
Number of rows-  6896317
Number of columns-  18
size of data-  124133706
Dimension of data-  2
```

```
In [ ]: print('shape of data: ',data.shape)
```

```
shape of data:  (6896317, 18)
```

```
In [ ]: data.dtypes
```

```
Out[6]: VendorID                   float64
        tpep_pickup_datetime        object
        tpep_dropoff_datetime       object
        passenger_count            float64
        trip_distance              float64
        RatecodeID                 float64
        store_and_fwd_flag          object
        PULocationID                 int64
        DOLocationID                 int64
        payment_type               float64
        fare_amount                float64
        extra                      float64
        mta_tax                    float64
        tip_amount                 float64
        tolls_amount               float64
        improvement_surcharge      float64
        total_amount               float64
        congestion_surcharge       float64
        dtype: object
```

In [ ]:
```
print('columns of data - ')
for column in data.columns:
  print(column)
```

```
columns of data - 
VendorID
tpep_pickup_datetime
tpep_dropoff_datetime
passenger_count
trip_distance
RatecodeID
store_and_fwd_flag
PULocationID
DOLocationID
payment_type
fare_amount
extra
mta_tax
tip_amount
tolls_amount
improvement_surcharge
total_amount
congestion_surcharge
```

In [ ]:
```
print('Index of data-',data.index)
```

```
Index of data- RangeIndex(start=0, stop=6896317, step=1)
```

Conclusion

- In this data, there are 6896317 rows and 18 columns.
- The columns names of the data are VendorID, tpep_pickup_datetime, tpep_dropoff_time, passenger_count, trip_distance, RatecodeID, store_and_fwd_flag, PULocationID, DOLocationID, payment_type, fare_amount, extra, mta_tax, tip_amount, tolls_amount, improvement_surcharge, total_amount and congestion_surcharge.
- In this data, the fields or columns contain 13 float data type column, 2 integer data type, and 3 object data type columns.
- The index of data is start from zero(0) and end at 6896316.

1.4 File structure and content

Here, we see the structure of data, sample of data, and some statistic of data.

```
In [ ]: data.describe()
```

Out[9]:

	VendorID	passenger_count	trip_distance	RatecodeID	PULocationID	DOLocationID	payment_type	fare_amount	extra	
count	6.845299e+06	6.845299e+06	6.896317e+06	6.845299e+06	6.896317e+06	6.896317e+06	6.845299e+06	6.896317e+06	6.896317e+06	6.89
mean	1.666457e+00	1.550877e+00	2.973421e+00	1.065756e+00	1.636525e+02	1.614892e+02	1.298993e+00	1.359027e+01	1.108518e+00	4.92
std	4.714787e-01	1.174330e+00	1.643113e+01	9.309869e-01	6.605758e+01	7.033753e+01	4.879401e-01	1.522692e+02	1.259892e+00	7.23
min	1.000000e+00	0.000000e+00	-3.726453e+04	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	-1.472000e+03	-4.500000e+00	-5
25%	1.000000e+00	1.000000e+00	9.600000e-01	1.000000e+00	1.250000e+02	1.120000e+02	1.000000e+00	6.500000e+00	0.000000e+00	5.00
50%	2.000000e+00	1.000000e+00	1.600000e+00	1.000000e+00	1.620000e+02	1.620000e+02	1.000000e+00	9.500000e+00	5.000000e-01	5.00
75%	2.000000e+00	2.000000e+00	3.030000e+00	1.000000e+00	2.330000e+02	2.340000e+02	2.000000e+00	1.543000e+01	2.500000e+00	5.00
max	2.000000e+00	9.000000e+00	1.913018e+04	9.900000e+01	2.650000e+02	2.650000e+02	5.000000e+00	3.984684e+05	9.006000e+01	3.30

```
In [ ]: data.head()
```

Out[10]:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOLocatio
0	1.0	2019-12-01 00:26:58	2019-12-01 00:41:45	1.0	4.2	1.0	N	142	
1	1.0	2019-12-01 00:12:08	2019-12-01 00:12:14	1.0	0.0	1.0	N	145	
2	1.0	2019-12-01 00:25:53	2019-12-01 00:26:04	1.0	0.0	1.0	N	145	
3	1.0	2019-12-01 00:12:03	2019-12-01 00:33:19	2.0	9.4	1.0	N	138	
4	1.0	2019-12-01 00:05:27	2019-12-01 00:16:32	2.0	1.6	1.0	N	161	

In [ ]:
```
data.sample(10)
```

Out[11]:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOI
4337748	2.0	2019-12-18 16:44:27	2019-12-18 17:57:40	1.0	10.24	1.0	N	138	
2931379	2.0	2019-12-13 00:34:22	2019-12-13 00:37:56	3.0	0.50	1.0	N	234	
1302092	1.0	2019-12-06 15:52:12	2019-12-06 16:07:32	1.0	1.70	1.0	N	162	
2694256	2.0	2019-12-12 08:45:23	2019-12-12 08:50:51	1.0	1.49	1.0	N	75	
4175230	1.0	2019-12-17 22:04:14	2019-12-17 22:09:55	2.0	1.20	1.0	N	143	
6562239	2.0	2019-12-30 12:17:29	2019-12-30 12:27:08	3.0	1.13	1.0	N	162	
1580307	1.0	2019-12-07 16:53:47	2019-12-07 17:42:33	1.0	7.90	1.0	N	90	
2511804	1.0	2019-12-11 15:50:13	2019-12-11 16:01:25	1.0	2.70	1.0	N	262	
3284524	2.0	2019-12-14 11:23:40	2019-12-14 11:40:42	1.0	3.15	1.0	N	79	
2953185	2.0	2019-12-13 04:51:08	2019-12-13 05:20:52	5.0	16.64	2.0	N	161	

In [ ]:
```
data.tail()
```

Out[12]:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOI
6896312	NaN	2019-12-31 00:07:00	2019-12-31 00:46:00	NaN	12.78	NaN	NaN	230	
6896313	NaN	2019-12-31 00:20:00	2019-12-31 00:47:00	NaN	18.52	NaN	NaN	219	
6896314	NaN	2019-12-31 00:50:00	2019-12-31 01:21:00	NaN	13.13	NaN	NaN	161	
6896315	NaN	2019-12-31 00:38:19	2019-12-31 01:19:37	NaN	14.51	NaN	NaN	230	
6896316	NaN	2019-12-31 00:21:00	2019-12-31 00:56:00	NaN	-17.16	NaN	NaN	193	

Conclusion

- VendorID contain only two values 1.0 and 2.0, it indicate that there are two taxi companies in New York city.
- The taxi can provide services minimum 1 person and maximum 9 person at a time.

- There are five method to pay amount for the services by customer to taxi driver
- The field store_and_fwd_flag contain only "N" and "Y". "N" indicates that the trip data was sent immediately and "Y" indecates that held in the memory of taxi.
- The taxi always available either day or night.
- The data have missing value which is not good for analysiss of the data.

1.5 Missing value

Missing value is important to know of the data because it create many issues for extracting useful information, good analysis of data, creation of best model and so on. Many modelling procedures break down when missing values are involved and the corresponding rows will either have to be removed completely or the values need to be estimated somehow.

If data have missing values then remove or fill those missing value accoring to data or challenge or goal with data.

In [ ]:
```
data.isnull().sum()
```

Out[13]:
```
VendorID                 51018
tpep_pickup_datetime         0
tpep_dropoff_datetime        0
passenger_count          51018
trip_distance                0
RatecodeID               51018
store_and_fwd_flag       51018
PULocationID                 0
DOLocationID                 0
payment_type             51018
fare_amount                  0
extra                        0
mta_tax                      0
tip_amount                   0
tolls_amount                 0
improvement_surcharge        0
total_amount                 0
congestion_surcharge         0
dtype: int64
```

```
In [ ]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6896317 entries, 0 to 6896316
Data columns (total 18 columns):
 #   Column                 Dtype  
---  ------                 -----  
 0   VendorID               float64
 1   tpep_pickup_datetime   object 
 2   tpep_dropoff_datetime  object 
 3   passenger_count        float64
 4   trip_distance          float64
 5   RatecodeID             float64
 6   store_and_fwd_flag     object 
 7   PULocationID           int64  
 8   DOLocationID           int64  
 9   payment_type           float64
 10  fare_amount            float64
 11  extra                  float64
 12  mta_tax                float64
 13  tip_amount             float64
 14  tolls_amount           float64
 15  improvement_surcharge  float64
 16  total_amount           float64
 17  congestion_surcharge   float64
dtypes: float64(13), int64(2), object(3)
memory usage: 947.1+ MB
```

```
In [ ]: data[data['VendorID'].isna()]
```

```
Out[15]:
```

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOI
6845299	NaN	2019-12-01 11:29:00	2019-12-01 12:17:00	NaN	18.21	NaN	NaN	107	
6845300	NaN	2019-12-01 11:40:42	2019-12-01 12:00:04	NaN	7.92	NaN	NaN	263	
6845301	NaN	2019-12-01 11:00:00	2019-12-01 11:37:00	NaN	9.38	NaN	NaN	22	
6845302	NaN	2019-12-01 11:43:00	2019-12-01 12:16:00	NaN	11.39	NaN	NaN	121	
6845303	NaN	2019-12-01 11:01:35	2019-12-01 11:17:58	NaN	3.01	NaN	NaN	75	
...	...	...	...	...	...	...	...	...	
6896312	NaN	2019-12-31 00:07:00	2019-12-31 00:46:00	NaN	12.78	NaN	NaN	230	
6896313	NaN	2019-12-31 00:20:00	2019-12-31 00:47:00	NaN	18.52	NaN	NaN	219	
6896314	NaN	2019-12-31 00:50:00	2019-12-31 01:21:00	NaN	13.13	NaN	NaN	161	
6896315	NaN	2019-12-31 00:38:19	2019-12-31 01:19:37	NaN	14.51	NaN	NaN	230	
6896316	NaN	2019-12-31 00:21:00	2019-12-31 00:56:00	NaN	-17.16	NaN	NaN	193	

51018 rows × 18 columns

```
In [ ]: data[(data['VendorID'].isna()) & (data['passenger_count'].isna()) & (data['RatecodeID'].isna()) & (data['store_and_fwd_f
```

Out[16]:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOI
6845299	NaN	2019-12-01 11:29:00	2019-12-01 12:17:00	NaN	18.21	NaN	NaN	107	
6845300	NaN	2019-12-01 11:40:42	2019-12-01 12:00:04	NaN	7.92	NaN	NaN	263	
6845301	NaN	2019-12-01 11:00:00	2019-12-01 11:37:00	NaN	9.38	NaN	NaN	22	
6845302	NaN	2019-12-01 11:43:00	2019-12-01 12:16:00	NaN	11.39	NaN	NaN	121	
6845303	NaN	2019-12-01 11:01:35	2019-12-01 11:17:58	NaN	3.01	NaN	NaN	75	
...	...	...	...	...	...	...	...	...	
6896312	NaN	2019-12-31 00:07:00	2019-12-31 00:46:00	NaN	12.78	NaN	NaN	230	
6896313	NaN	2019-12-31 00:20:00	2019-12-31 00:47:00	NaN	18.52	NaN	NaN	219	
6896314	NaN	2019-12-31 00:50:00	2019-12-31 01:21:00	NaN	13.13	NaN	NaN	161	
6896315	NaN	2019-12-31 00:38:19	2019-12-31 01:19:37	NaN	14.51	NaN	NaN	230	
6896316	NaN	2019-12-31 00:21:00	2019-12-31 00:56:00	NaN	-17.16	NaN	NaN	193	

51018 rows × 18 columns

```
In [ ]: data = data.dropna()
```

```
In [ ]: print(data.shape)
```

```
(6845299, 18)
```

```
In [ ]: data.isnull().sum()
```

```
Out[19]: VendorID                 0
         tpep_pickup_datetime     0
         tpep_dropoff_datetime    0
         passenger_count          0
         trip_distance            0
         RatecodeID               0
         store_and_fwd_flag       0
         PULocationID             0
         DOLocationID             0
         payment_type             0
         fare_amount              0
         extra                    0
         mta_tax                  0
         tip_amount               0
         tolls_amount             0
         improvement_surcharge    0
         total_amount             0
         congestion_surcharge     0
         dtype: int64
```

Conclusion

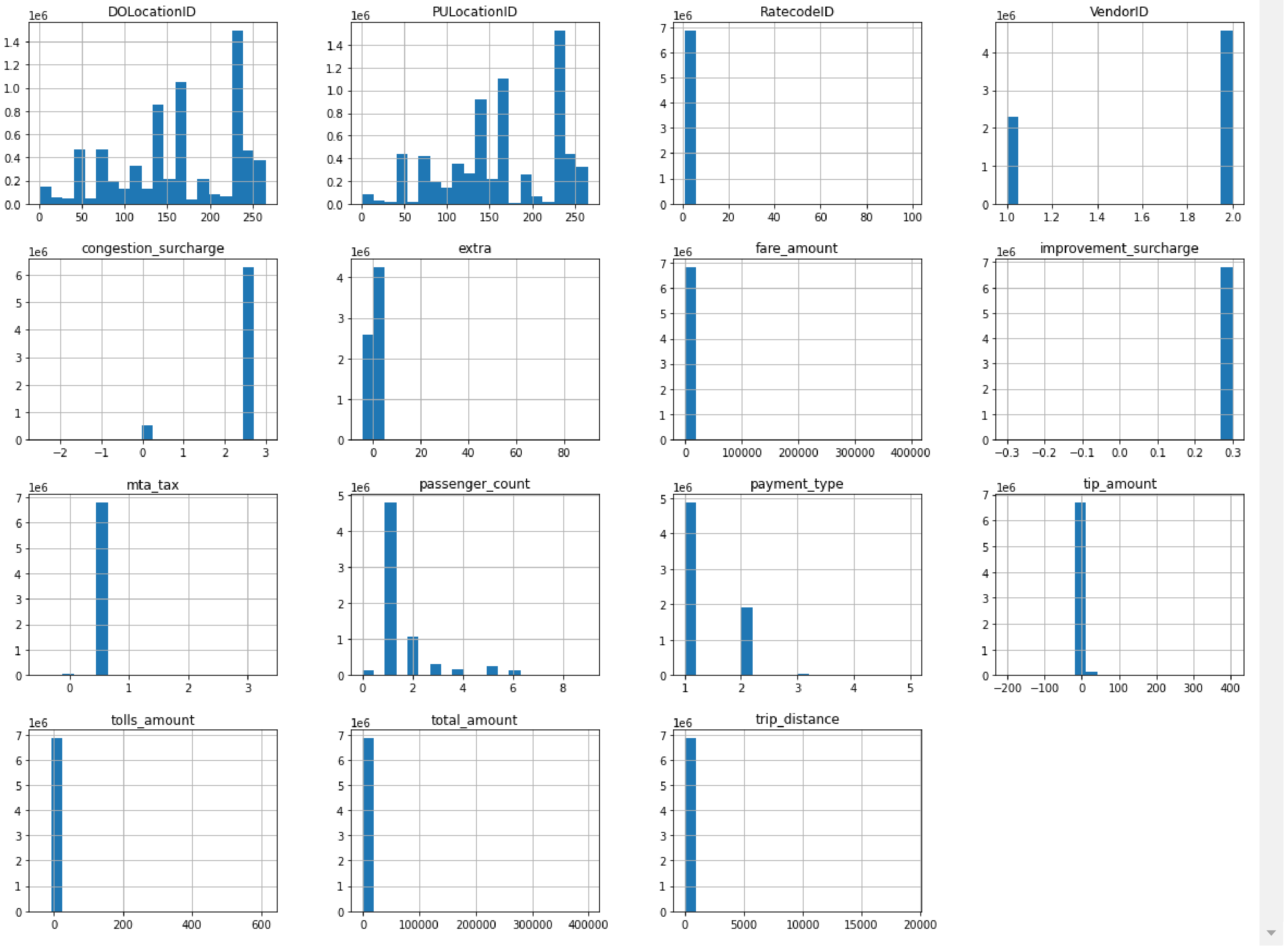
- In this data set, there are 51018 rows which have missing data.
- There are five columns data with missing values.
- The columns name are VendorID, passenger_count, store_and_fwd_flag, RatecodeID and payment type which have missing value.
- As meintioned above all 51018 rows have same missing data.
- We remove all missing value of the data beacuse these are very big data if I remove some small data then the EDA and models both are performed better with more accuracy.

2 Indiviual feature Visualization

Histogram of the data

- Histogram is graphical representation of data.
- It work on only numerical data.
- It is used to get overview of field of data which data types are int or float.

In [ ]:
```
data.hist(bins=20,figsize=(20,15));
```

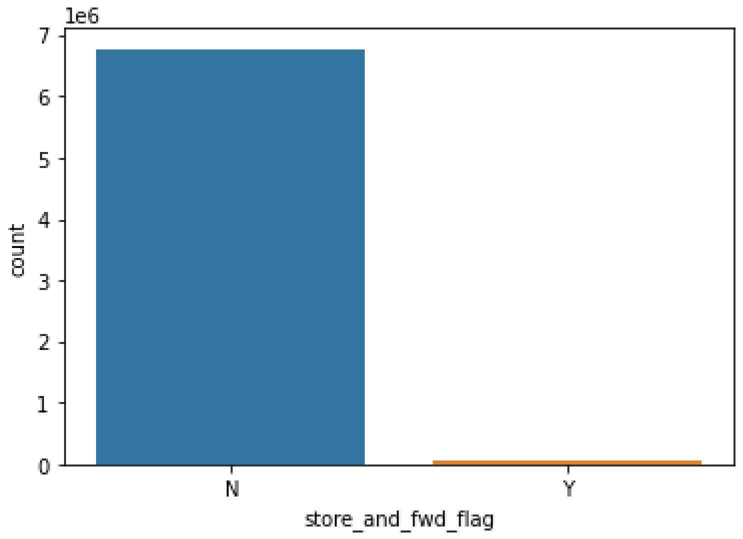

DOLocationID
PULocationID
RatecodeID
VendorID
congestion_surcharge
extra
fare_amount
improvement_surcharge
mta_tax
passenger_count
payment_type
tip_amount
tolls_amount
total_amount
trip_distance
1e6

Conclusion

- In this graph, there are 15 subplots according to field of data.
- The graph explain the over view of each field where data are number.

2.1 Store_and_fwd_flag graph

```
In [ ]: sns.countplot(x='store_and_fwd_flag', data=data);
```



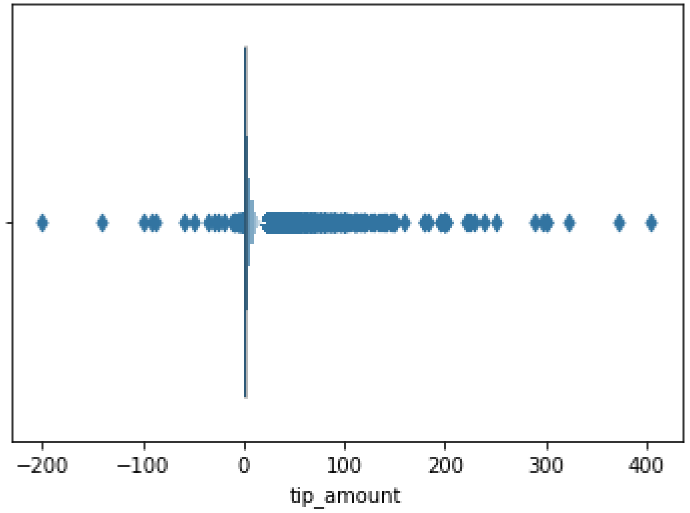
Conclusion

- This graph indicate that the maximum trip data was sent immediately to the server using app or website.
- When the app or website or server is not work properly, the taxi driver hold the data. These data can create inconsistency or redundency in the data on the server.
- It indicates that your server and app or wesite are working properly.
- The data is easily maintain by the server

2.2 Tip amount

```
In [ ]: sns.boxenplot(data['tip_amount'])
```

```
Out[67]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb3618f5438>
```



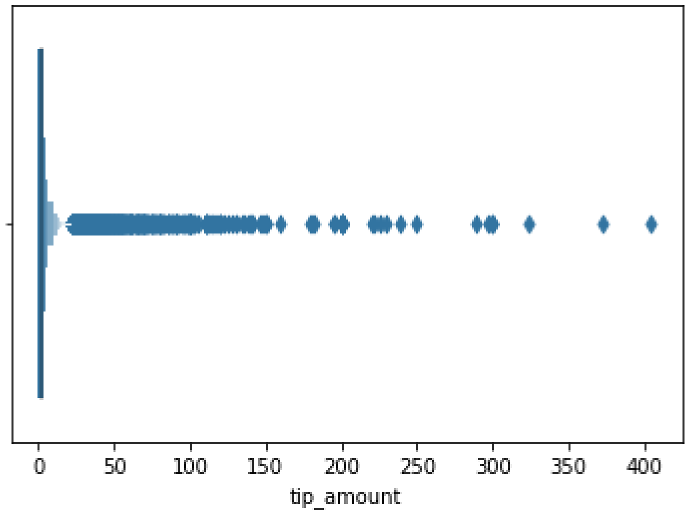
- In this graph shows that tip amount is negative but according to sense tip amount cannot be negative.
- We need to remove that instance of data which have negative tip amount.

```
In [ ]: index_tip_amount_negative=data[data['tip_amount']<0].index
```

```
In [ ]: data = data.drop(index_tip_amount_negative,axis=0)
```

In [ ]:
```
sns.boxenplot(data['tip_amount'])
```

Out[74]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb3999a5940>
```



```
In [ ]: data['tip_amount'].describe()
```

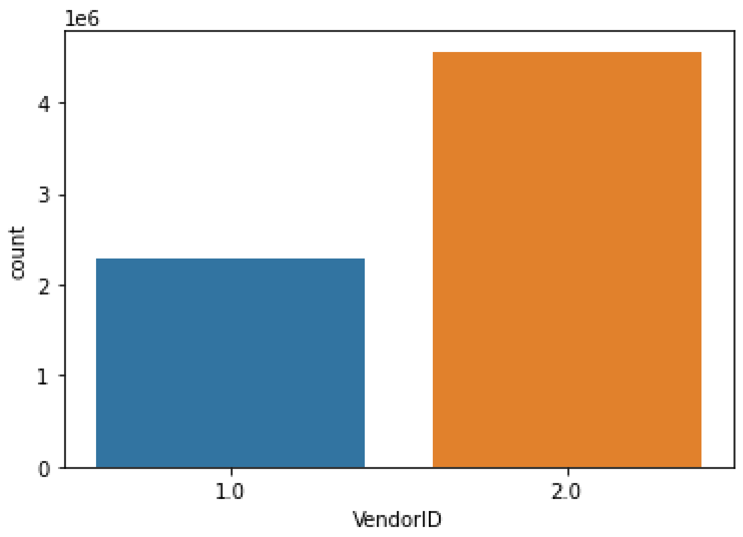
```
Out[75]: count    6.845096e+06
         mean     2.261029e+00
         std      2.875812e+00
         min      0.000000e+00
         25%      0.000000e+00
         50%      1.950000e+00
         75%      3.000000e+00
         max      4.044400e+02
         Name: tip_amount, dtype: float64
```

Conclusion

- The tip amount is not depend on any feature. The tip amount will be given when passenger are happy.
- There are no limit of tip amount and according to data analysis the maximum tip amount is 400 USD and minimum tip amount is 0 USD.
- The maximum number of passenger did not give any amount of tip to taxi driver.

2.3 VendorID graph

```
In [ ]: sns.countplot(x='VendorID',data=data);
```



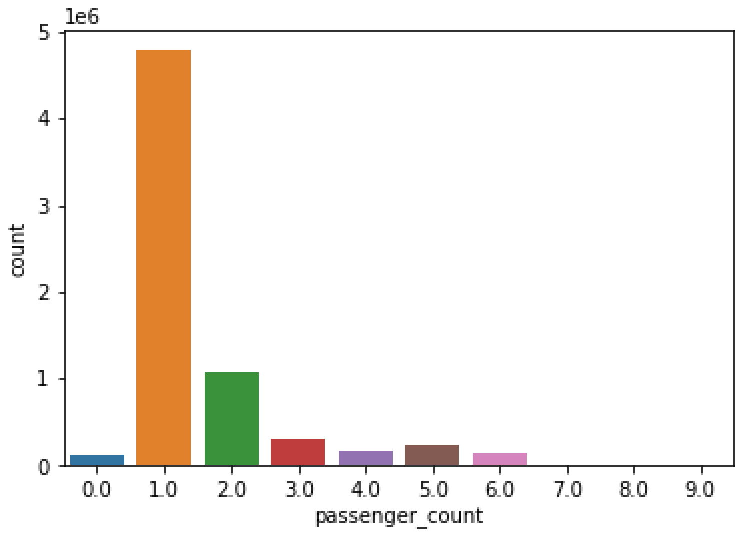
Conclusion

- There are two vendorID 1.0 and 2.0. it indicates two companies.
- The 2nd company (2.0) take more pickup and dropoff to the customer.
- The graph shows that the 2nd compnay(2.0) is more larger than 1st company(1.0).
- The 2nd company provide better services than 1st company so that the profit of 2nd company is higher than 1st company.

2.4 Passenger_count graph

```
In [ ]: sns.countplot(x='passenger_count',data=data)
```

```
Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc3a5cd09e8>
```

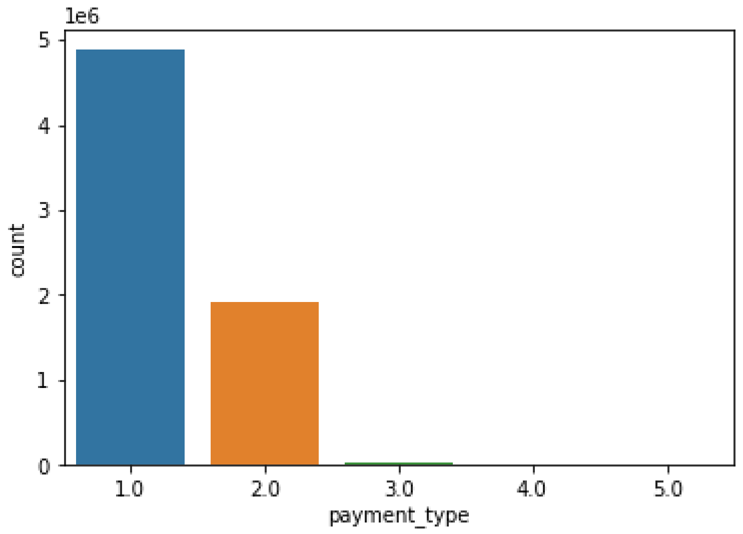


Conclusion

- This graph indicate that the customer want to travel alone.
- The most common customers are worker so that every customer want to get services first.
- Driver of taxi can handle maximum nine customer at a time.
- The families are travel in the weekends.

2.5 Payment_type graph

```
In [ ]: sns.countplot(x='payment_type',data=data);
```



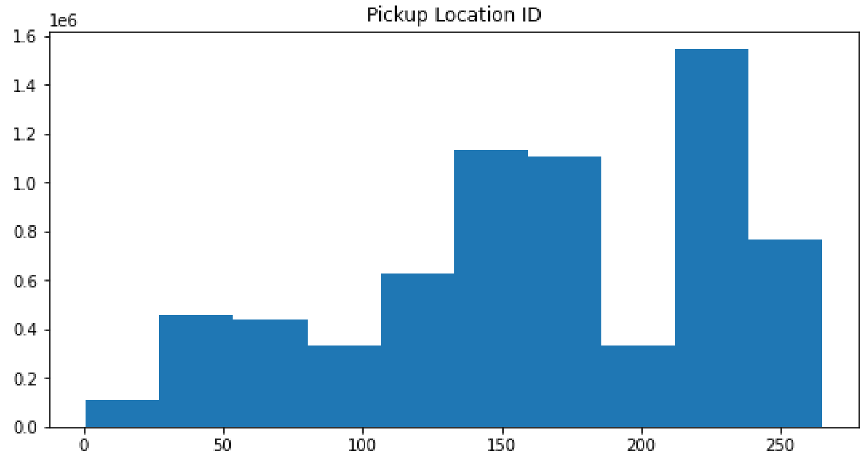
Conclusion

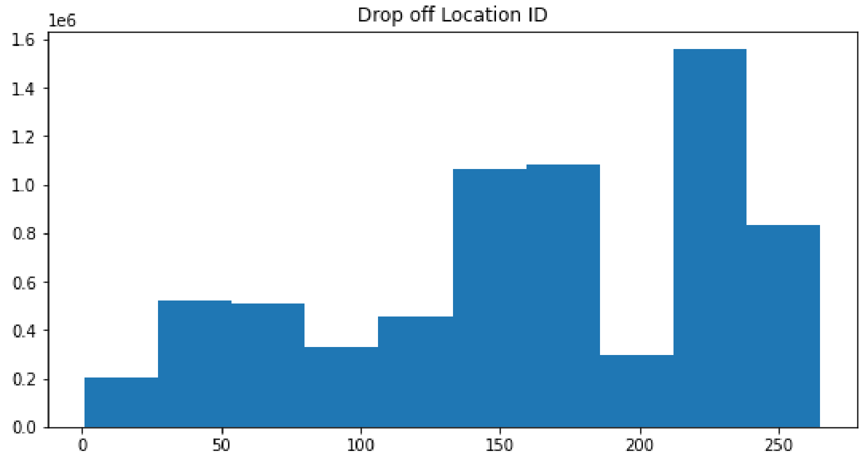
- There are five type of payment methods as follow-
  - By cash
  - By Card (Debit or Credit card)
  - By Paytm
  - By GooglePay
  - By PhonePay
- The maximum number of passenger pay amount for the services to companies by cash or by card (credit or debit card).

2.6 Pickup location ID and Drop off location ID

```
In [ ]: plt.figure(figsize=(20,10))
        plt.subplot(2,2,1)
        plt.hist(data['PULocationID'])
        plt.title('Pickup Location ID')
        plt.subplot(2,2,2)
        plt.hist(data['DOLocationID'])
        plt.title('Drop off Location ID')
```

```
Out[15]: Text(0.5, 1.0, 'Drop off Location ID')
```





Conclusion

- This graphs indicates that the pickup location ID and Drop off location ID both are approximate same. It means that the passenger are travel in limited area.
- This pickup locations indicate societal area and the drop off loacations indicate the industrial area.
- The distance between pickup location ID's and Drop off location ID's are minor or small.

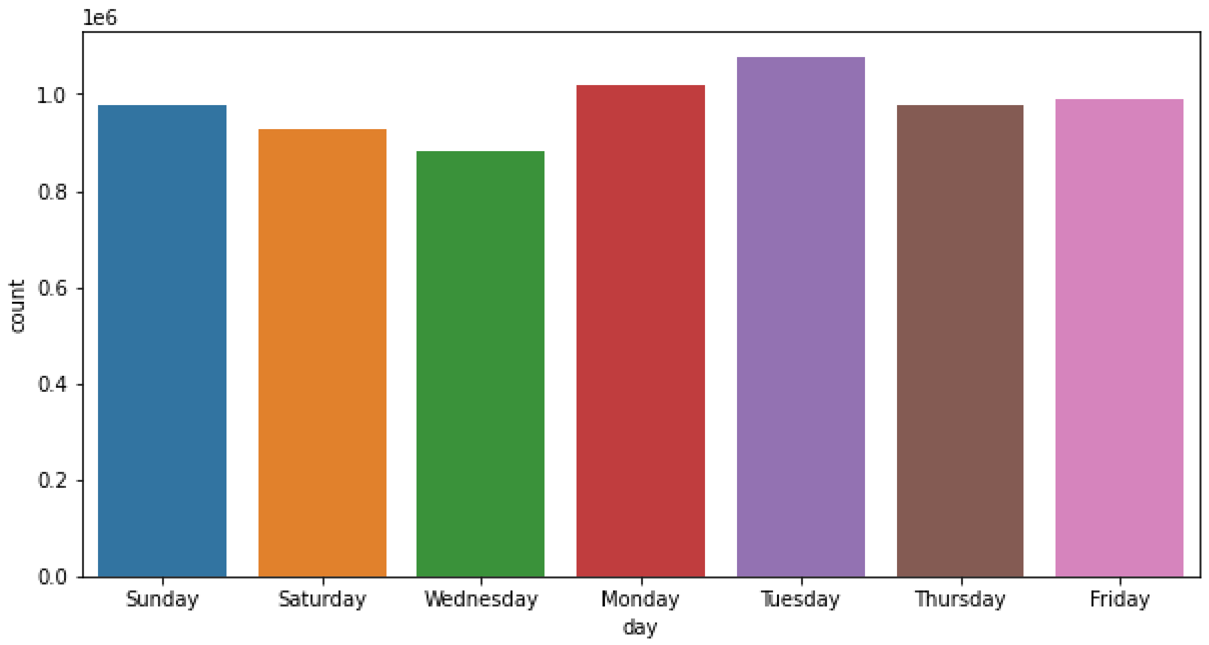
2.7 Day graph

```
In [ ]: data['tpep_dropoff_datetime'] = pd.to_datetime(data['tpep_dropoff_datetime'])
        data['tpep_pickup_datetime'] = pd.to_datetime(data['tpep_pickup_datetime'])
```

```
In [ ]: fun = lambda x:x.day_name()
        data['day']=data['tpep_pickup_datetime'].apply(fun)
```

```
In [ ]: plt.figure(figsize=(10,5))
        sns.countplot(x='day',data=data)
```

```
Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc357cbdfd0>
```



Conclusion

- Here, we can see that the maximum number of passenger use taxi on **Monday and Tuesday**.

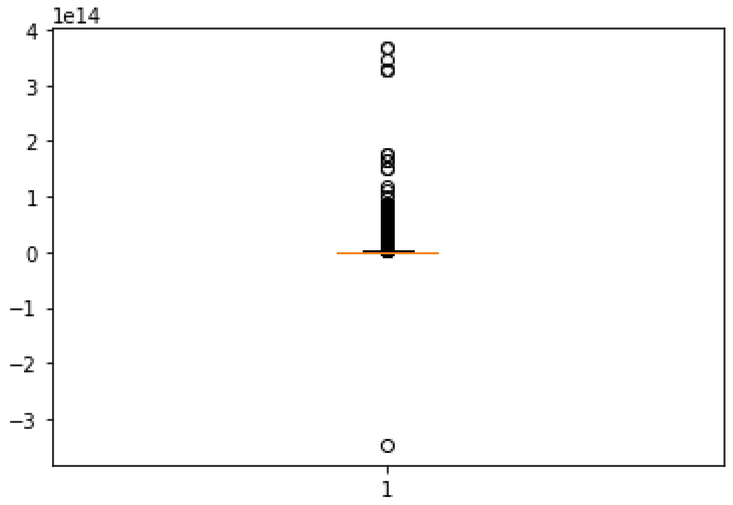
2.8 Time duration graph

Here, we calculate the time duration until the person is in taxi. We plot the graph of time duration

The time duration will be difference of drop off time and pickup time of the passenger.

```
In [ ]: data['time_duration']=data['tpep_dropoff_datetime']-data['tpep_pickup_datetime']
```

```
In [ ]: plt.boxplot(data['time_duration']);
```



- We find that one instance of time duration is negative it means there are some wrong information. Because the drop off time always remain greater than pickup time. So that the time duration always positive.
- We need to remove this type of instance where the drop off time is lower than pickup time.

```
In [ ]: print(data['time_duration'].max()) # Maximum time duration
        print(data['time_duration'].min()) # Minimum time duration
```

```
4 days 05:54:47
-5 days +23:40:32
```

```
In [ ]: np.sum(data['tpep_dropoff_datetime']< data['tpep_pickup_datetime']) # number of instance where the pickup time is greate
```

```
Out[30]: 1
```

```
In [ ]: data[data['tpep_dropoff_datetime']<data['tpep_pickup_datetime']]  #Those row where drop off time is low.
```

Out[31]:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DO
2036290	1.0	2019-12-09 15:53:10	2019-12-05 15:33:42	1.0	3.7	1.0	N	186	

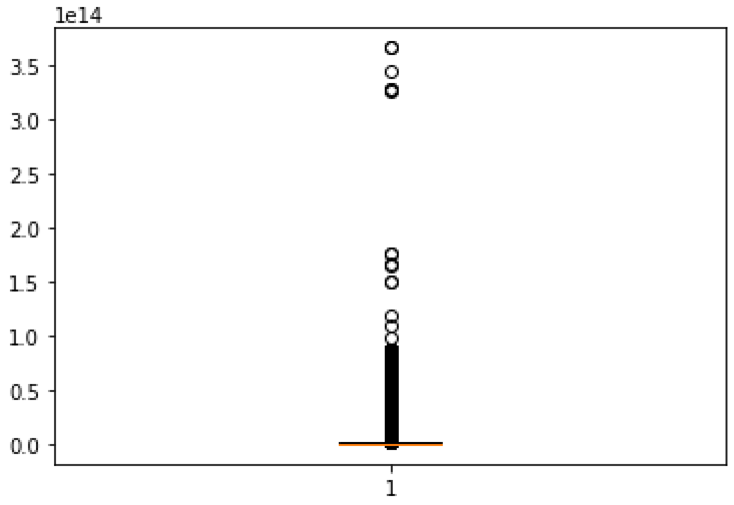
```
In [ ]: data=data.drop(2036290,axis=0)
```

```
In [ ]: data.head()
```

Out[33]:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOLocatio
0	1.0	2019-12-01 00:26:58	2019-12-01 00:41:45	1.0	4.2	1.0	N	142	
1	1.0	2019-12-01 00:12:08	2019-12-01 00:12:14	1.0	0.0	1.0	N	145	
2	1.0	2019-12-01 00:25:53	2019-12-01 00:26:04	1.0	0.0	1.0	N	145	
3	1.0	2019-12-01 00:12:03	2019-12-01 00:33:19	2.0	9.4	1.0	N	138	
4	1.0	2019-12-01 00:05:27	2019-12-01 00:16:32	2.0	1.6	1.0	N	161	

In [ ]:
```
plt.boxplot(data['time_duration']);
```



```
In [ ]: data['time_duration'].describe()
```

```
Out[35]: count                        6845298
         mean      0 days 00:18:28.104512
         std       0 days 01:10:33.984347
         min              0 days 00:00:00
         25%              0 days 00:06:47
         50%              0 days 00:11:37
         75%              0 days 00:19:18
         max              4 days 05:54:47
         Name: time_duration, dtype: object
```

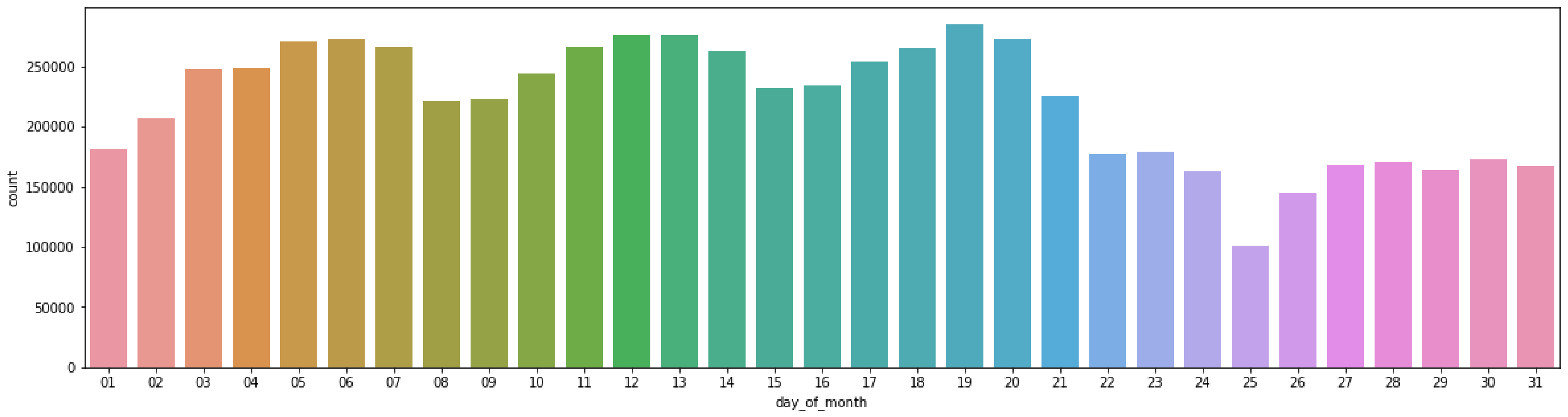
Conclusion

- The maximum time duration indicate that for how much time the passenger were on trip and those passenger may be visitor or tourist person.
- The minimum time indicate that the distance of pickup location and drop off location is minimum.
- For most of passenger, time duration of travelling lies between 11 minutes to 20 minutes.

2.9 Day of month graph

```
In [ ]: fun = lambda x:x.strftime('%d')
        data['day_of_month']= data['tpep_pickup_datetime'].apply(fun)
```

```
In [ ]: plt.figure(figsize=(20,5))
        sns.countplot(x='day_of_month', data=data);
```



Conclusion

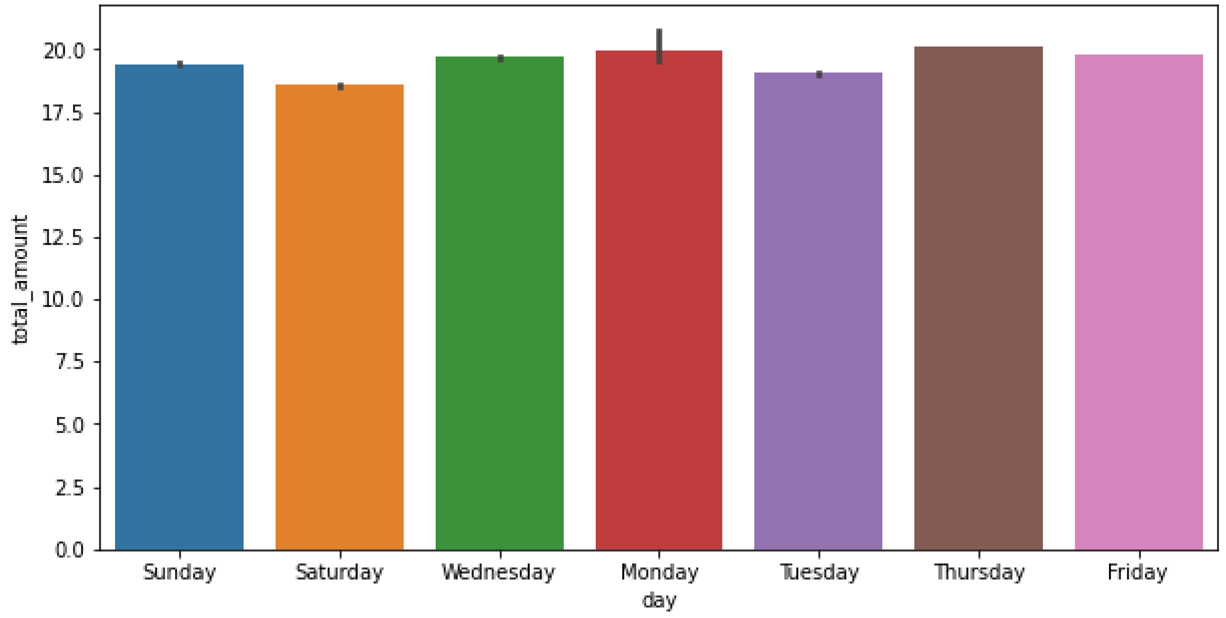
- In this graph, we can see clearly that in the weekends (saturday and sunday), most of customers taking rest so that taxi services will be provided to the minimum number of customers.
- The last week of December, everyone is busy in **Christmas day and new year party** so that the graph of number of pickup is low rather than other day or normal day.
- On **Christmas Day**(25 December 2019), there are few people travel on that day and more people are praying to God for blessing and happiness.

3 Feature relation

3.1 Day Vs total_amount

```
In [ ]: plt.figure(figsize=(10,5))
        sns.barplot(x='day',y='total_amount',data=data)
```

```
Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb4035fa908>
```

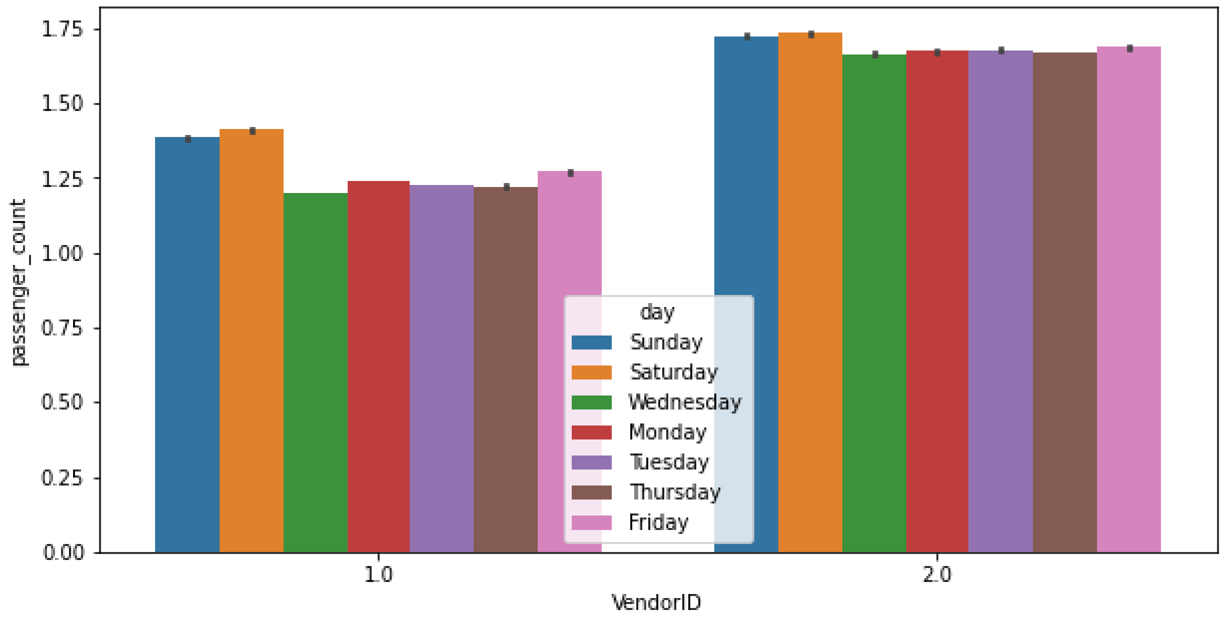


Conclusion

- This graph show that total amount which include fare_amount, mta_tax, extra charge, tip_amount,toll_amount and improvement charges are varies between 17.5 USD to 20 USD for whole week.
- The total amount is maximum on monday because everyone feel fresh and want go for work.
- The total amount is depend on days of the week.

3.2 VendorID Vs Passenger_count Vs Days

```
In [ ]: plt.figure(figsize=(10,5))
        sns.barplot(x='VendorID',y='passenger_count',hue='day',data=data);
```



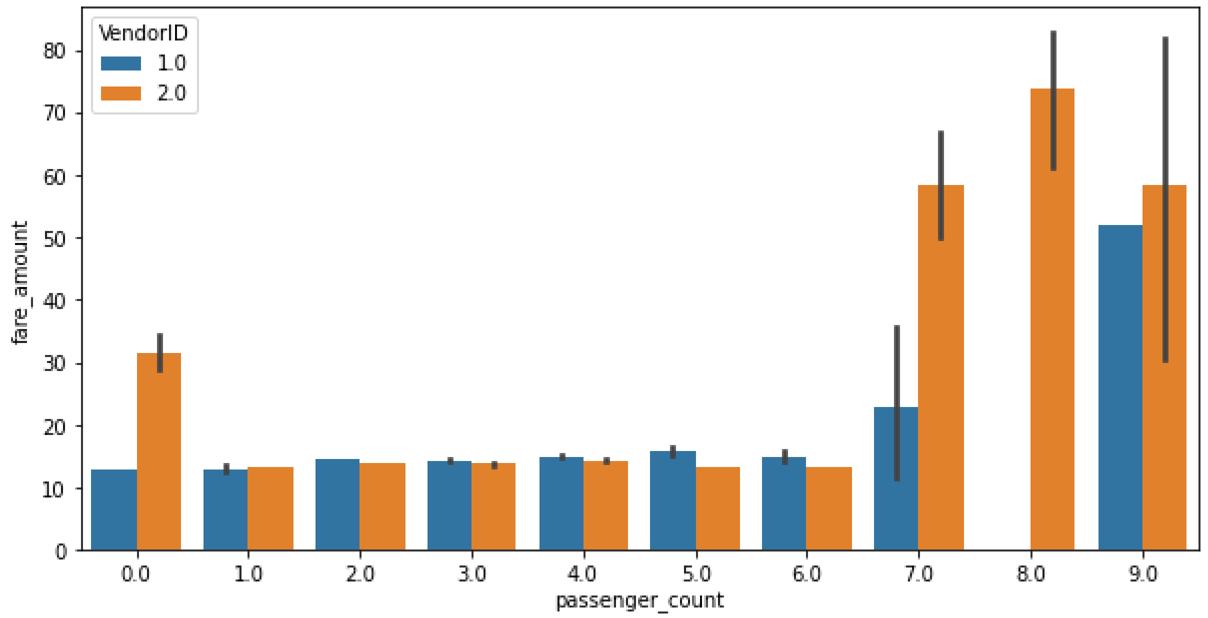
Conclusion

- On saturday and sunday, the maximum families or cuples want to spend time together so that number of passenger avail taxi services on weekend as compare to normal days.
- The graph shows that the 2nd company are handle more passenger than 1st company in all day.

3.3 Passenger_count Vs Fare_amount with VendorID

```
In [ ]: plt.figure(figsize=(10,5))
        sns.barplot(x='passenger_count', y='fare_amount',hue='VendorID',data=data)

Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb3765c3c88>
```



Conclusion

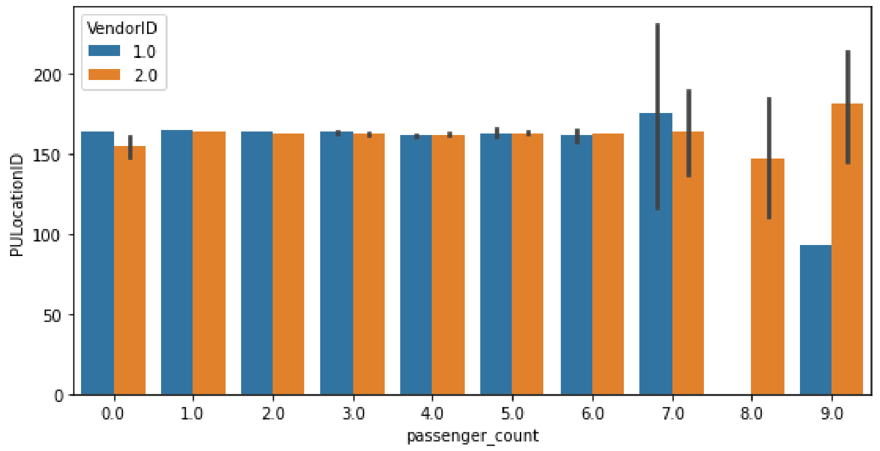
- In this bar plot, we can see that avrage fare_amount of 2nd company(VendorID 2.0) is high.
- The 2nd company provide more facility to passenger than 1st company.
- When the number of passenger are increase than fare amount will also increase.
- The 1st company not taking eight pasenger at a time.
- When the number of passenger are vary between one to six, the fare amount of both companies are same

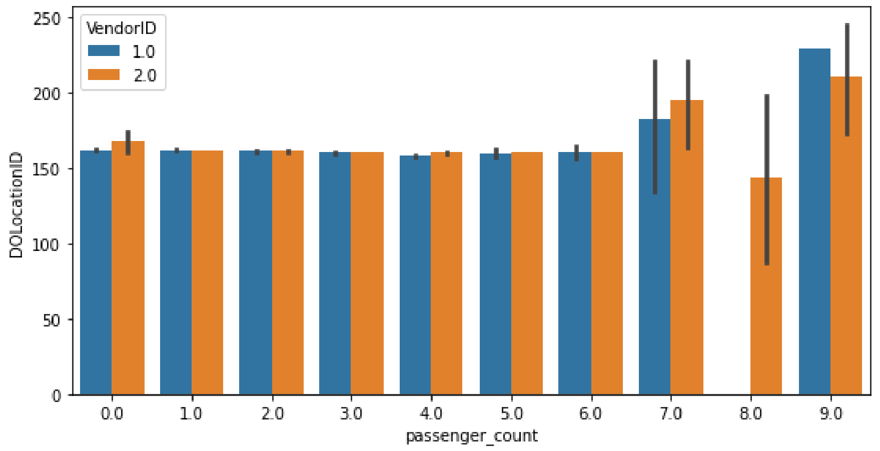
3.4 Passenger_count Vs Pickup location ID Vs VendorID

```
In [ ]: plt.figure(figsize=(20,10))
        plt.subplot(2,2,1)
        sns.barplot(x='passenger_count',hue='VendorID',y='PULocationID',data=data)

        plt.subplot(2,2,2)
        sns.barplot(x='passenger_count',hue='VendorID',y='DOLocationID',data=data)
```

```
Out[39]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb4181fc0f0>
```





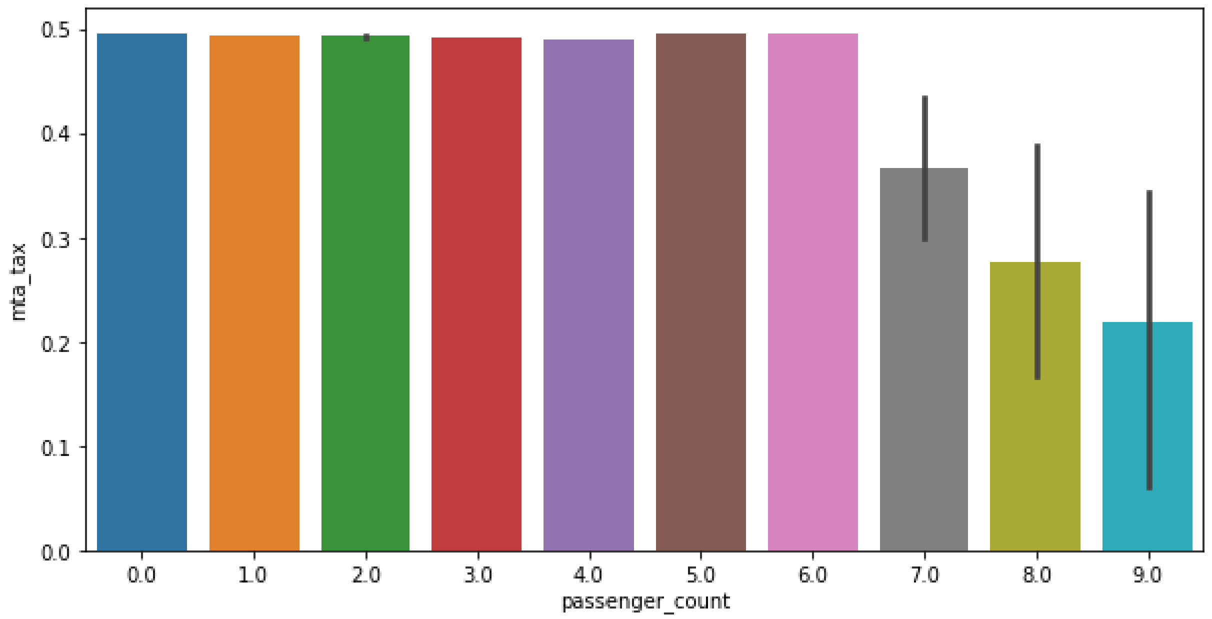
Conclusion

- The pickup and dropoff location ID are appoximate same when the number of passenger are zero to six.
- There are big societies from location ID 150 to location ID 155, so that the taxi get more passenger on these locations.
- There are large industrial area between 150 to 155 location ID.
- The location ID which are below than 100, these are outside area of the New York city. So that a taxi have more number of passenger in the outside area of the New York city as compare to within New York city.

3.5 Passenger_count Vs mta_tax

```
In [ ]: plt.figure(figsize=(10, 5))
        sns.barplot(data['passenger_count'], data['mta_tax'])
```

```
Out[44]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb418ae1358>
```



Conclusion

- When the number of passenger are zero to six, the mta_tax is same for both campany.
- The mta_tax is decrease with the increase of number of passenger at a time.
- The mta_tax is very low when the number of passenger is maximum.

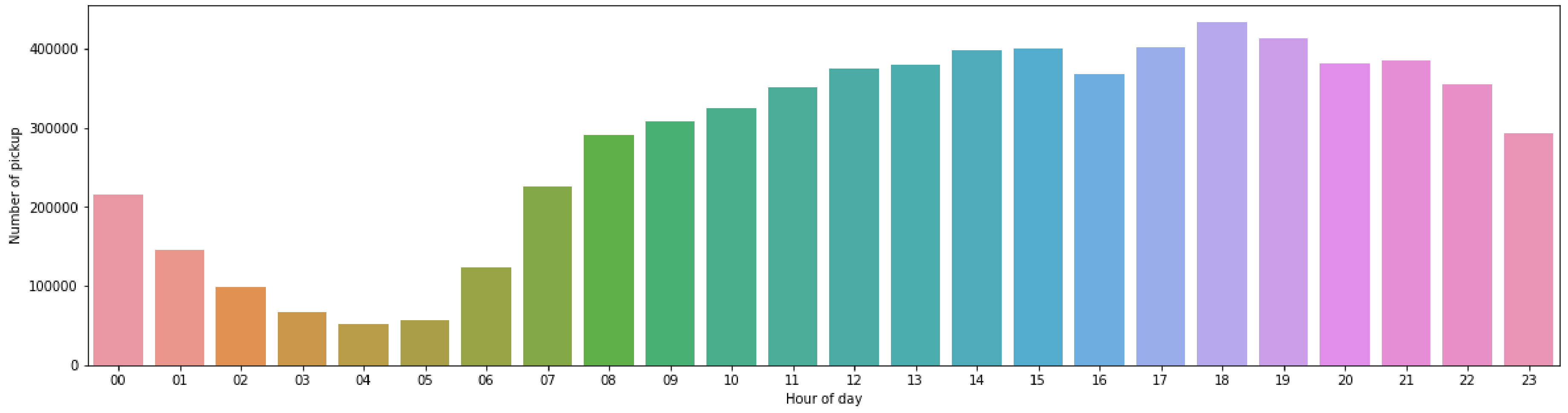
3.6 Number of pickup Vs Hours of day

```
In [ ]: fun = lambda x:x.strftime('%H')
        data['hours']= data['tpep_pickup_datetime'].apply(fun)
```

```
In [ ]: data['hours'].describe()
```

```
Out[59]: count      6845298
         unique          24
         top             18
         freq        433560
         Name: hours, dtype: object
```

```
In [ ]: plt.figure(figsize=(20,5))
        sns.countplot(data['hours'])
        plt.ylabel('Number of pickup')
        plt.xlabel('Hour of day')
        plt.show();
```



Conclusion

- The number of pickups decrese at night from 12:00 AM to 5:00 AM becuase there are minimum number taxi service available in the night.
- The number of pickups increase from 5:00 AM to 6:00PM because the maximum number of passenger will be available during this working hours.
- The number of pickup are maximum from 7:00 AM to 12:00 AM because it is opening time for the industries.

4 Correlation between the features

Correlation, as the name suggests, depicts a relationship between two or more features of the data.

```
In [ ]: corr = data.corr()
```

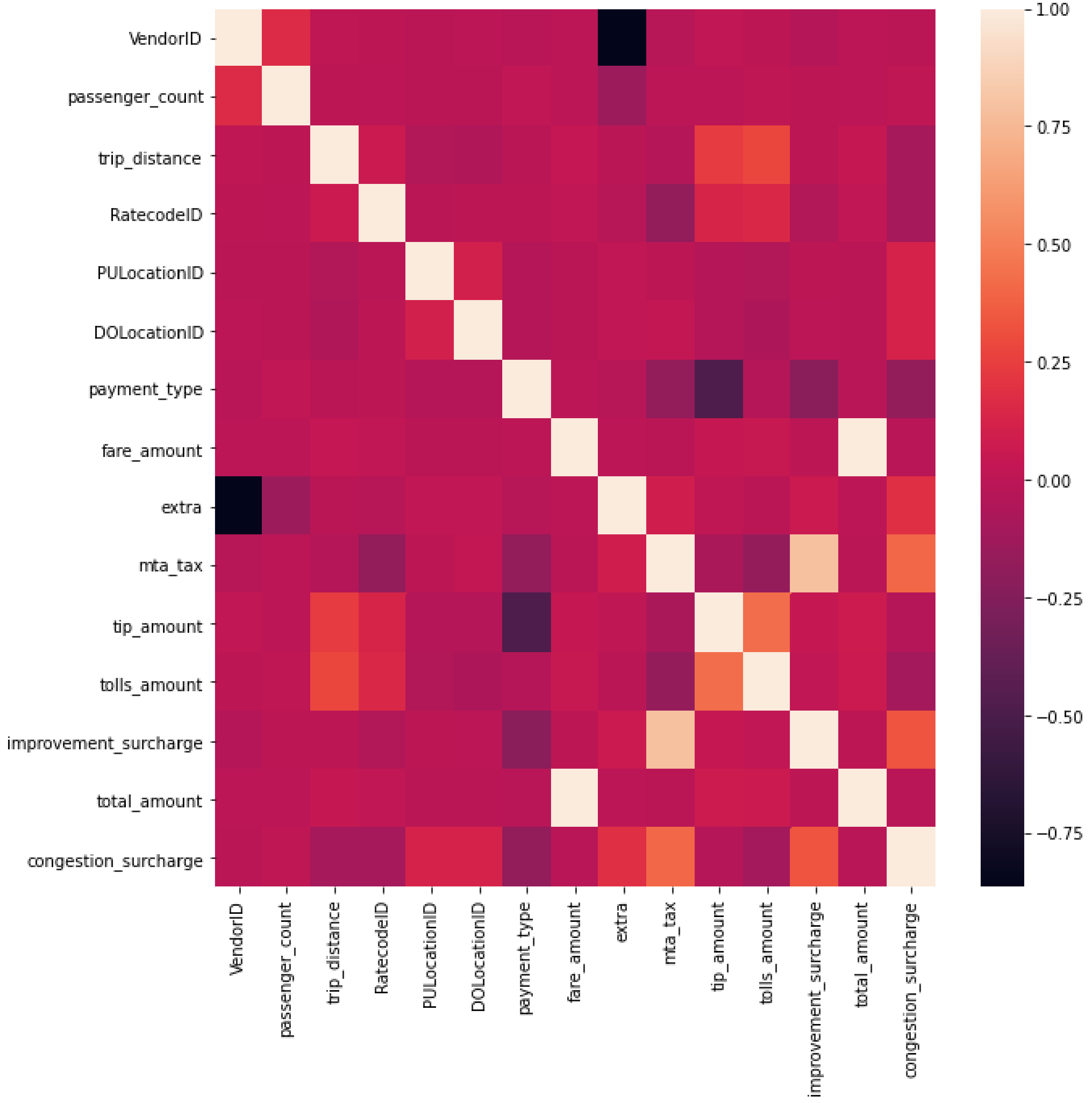
```
In [ ]: corr
```

Out[13]:

	VendorID	passenger_count	trip_distance	RatecodeID	PULocationID	DOLocationID	payment_type	fare_amount	extra
VendorID	1.000000	0.165504	0.011955	0.007823	-0.008160	-0.003435	-0.012721	0.000522	-0.864544
passenger_count	0.165504	1.000000	0.007955	-0.001556	-0.005576	-0.005497	0.018636	0.001092	-0.136908
trip_distance	0.011955	0.007955	1.000000	0.066688	-0.044534	-0.049477	-0.009172	0.032334	-0.005873
RatecodeID	0.007823	-0.001556	0.066688	1.000000	-0.006994	0.002849	0.005562	0.018238	-0.022250
PULocationID	-0.008160	-0.005576	-0.044534	-0.006994	1.000000	0.104698	-0.029042	-0.006184	0.022067
DOLocationID	-0.003435	-0.005497	-0.049477	0.002849	0.104698	1.000000	-0.030544	-0.006615	0.019757
payment_type	-0.012721	0.018636	-0.009172	0.005562	-0.029042	-0.030544	1.000000	-0.001947	-0.023440
fare_amount	0.000522	0.001092	0.032334	0.018238	-0.006184	-0.006615	-0.001947	1.000000	-0.001066
extra	-0.864544	-0.136908	-0.005873	-0.022250	0.022067	0.019757	-0.023440	-0.001066	1.000000
mta_tax	-0.026163	0.001165	-0.033941	-0.173856	0.003378	0.026107	-0.178160	-0.009042	0.090459
tip_amount	0.021828	-0.000519	0.242322	0.126870	-0.031530	-0.036662	-0.481547	0.045013	0.009617
tolls_amount	0.008094	0.011566	0.285936	0.149795	-0.047562	-0.063878	-0.033433	0.047690	-0.007440
improvement_surcharge	-0.039185	0.003415	0.006053	-0.043973	0.002209	0.000817	-0.222757	0.005404	0.065716
total_amount	0.000995	0.001308	0.039728	0.021758	-0.006796	-0.007481	-0.012450	0.999657	0.001202
congestion_surcharge	-0.009158	0.014328	-0.092883	-0.097568	0.121957	0.122296	-0.176097	-0.015785	0.188388

```
In [ ]: plt.figure(figsize=(10,10))
        sns.heatmap(corr)

Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x7f132fef04a8>
```


VendorID
passenger_count
trip_distance
RatecodeID
PULocationID
DOLocationID
payment_type
fare_amount
extra
mta_tax
tip_amount
tolls_amount
improvement_surcharge
total_amount
congestion_surcharge
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

Conclusion

- The negative correlation indicate the both features are inversely propotional. It means that if I increase value of one feature then the value of another feature will decrease and vice versa like-
    - VendorID and Extra (High negative correlation)
    - mta_tax and payment_type
- The positive correlation indicate the both features are directly propotional it means that if I increase value of one feature then the value of another feature will also increase like-
    - fare amount and total amount (High positive correlation)